bizzTainer web connector user guide

Version 1.1.0

© exxTainer Commerce GmbH, Germany

Contents

1 Introduction....3
2 Messages exchanged....3
3 Usage of the Functionalities....4
3.1 Access parameters....4
3.2 Functions for message download....4
3.3 Function for message upload....4
4 SOAP interface....5
4.1 SOAP service URL....5
4.2 Access parameters....5
4.3 Message transfer....5
4.3.1 getList....5
4.3.2 getMessage....6
4.3.3 setMessageOK....6
4.3.4 putMessage....7
4.4 Java example code....8
5 REST style interface....11
5.1 REST service URLs....11
5.2 Access parameters....11
5.3 Message transfer....11
5.3.1 getList....11
5.3.2 getMessage....12
5.3.3 putMessage....12
5.4 Java example code....12

®™ Namentlich genannte Hard- und Softwareprodukte sind Warenzeichen oder eingetragene Warenzeichen der jeweiligen Firmen.

1 Introduction

This document describes the usage of the bizzTainer web connectors to access a bizzTainer hub via standard HTTP and HTTPS protocols.

All messages exchanged via these interfaces must be conform to the openTRANS® standard version 1.0.

Two interfaces are provided:

- SOAP interface
- REST style interface

The access to these will be described in the following sections.

2 Messages exchanged

All messages exchanged via the bizzTainer web connectors need to conform to the openTRANS® standard version 1.0. See http://www.opentrans.org for further information on these formats.

The web connectors have no mapping functionalities – if you need this in order to connect to your inhouse system have a look at the bizzTainer connector which is especially designed for mapping an complex integration purposes (http://www.bizztainer.org).

The messages that can be exchanged with the bizzTainer hub are the openTRANS® message types

- RFQ – request for quotation
- QUOTATION – quotation
- ORDER – order
- ORDERRESPONSE – order response
- DISPATCHNOTIFICATION – dispatch notification
- INVOICE – invoice

If you connect to a bizzTainer hub via a the SOAP or REST style interface the messages you upload **MUST** conform to this standard – otherwise the upload will fail.

All messages downloaded are also in openTRANS® format.

3 Usage of the Functionalities

The bizzTainer web connector interfaces support four functions for the access to a bizzTainer hub.

The following sections describe the functions in general whereas the implementation specific details are described in the SOAP and REST sections.

All functions return an XML document as a response. The receiving application must evalute this for failures and success.

Please note that due to the SOAP messaging overhead the execution time of a SOAP function takes about twice the time of executing the corresponding REST style function.

3.1 Access parameters

All functions mentioned in the following two sections need authorization information in order to proceed. This means that you have to transfer the connector login information configured on the bizzTainer hub you want to connect to.

Your bizzTainer administrator has to activate the connector access and can set the parameters *connector-login* **(typically your bizzTainer-id) and** *connector-password***.**

You must add to all of the following functions your login *connector-login* and password *connector-password* otherwise the calls will fail with `not authorized`.

3.2 Functions for message download

The functions for message download are

1. `getList`
   This retrieves a list of available messages for download. The list containes the message type and the bizzTainer container-id.
2. `getMessage(messageType, btcid)`
   This downloads the message of type `messageType` from the bizzTainer container with the id `btcid`.
3. `setMessageOK(messageType, btcid)`
   If the download of the message to your local system was successful, this function needs to be called in order to set the message status on the bizzTainer hub correctly to *downloaded*.

   Do not forget this because otherwise the next call to getList will list you this message again and you may import this into your system again.

3.3 Function for message upload

The function for message upload is

`putMessage(message)`
This uploads the message `message` (XML in openTRANS® format) to the bizzTainer hub. If neccessary a new bizztainer container is created and (depending on your bizzTainer hub settings) the message is automatically transferred to the recipient.

4 SOAP interface

This section describes the SOAP specific parameters of the functions described in the previous section.

4.1 SOAP service URL

The bizzTainer SOAP service URL is

http://<your.bizztainer.host>/btp/connector/soap/service

where <your.bizztainer.host> is the URL of the bizzTainer hub you are connected to.

4.2 Access parameters

The login parameters are transferred as additional SOAP method parameters. See the WSDL description in
http://<your.bizztainer.host>/btp/connector/soap/service/wsdl
for further details.

Since the login parameters are transferred within the message, make sure that you use the encrypted HTTPS protocol for message exchange.

4.3 Message transfer

The following data describe the SOAP message transfer between the SOAP client and a bizzTainer hub.

4.3.1 getList

This is the client request to get the list of available messages:

```
POST /btp/connector/soap/service HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: 127.0.0.1
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 536


<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://ww-
w.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:getList soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://DefaultNamespace">
   <connectorLogin xsi:type="xsd:string">user1</connectorLogin>
   <connectorPassword xsi:type="xsd:string">user1</connectorPassword>
  </ns1:getList>
 </soapenv:Body>
</soapenv:Envelope>
```

which returns the following response

```
HTTP/1.1 200 OK
Set-Cookie: JSESSIONID=F8A5FB90FA3998BDF8E5CC59546940DC; Path=/axis
Content-Type: text/xml;charset=utf-8
Date: Fri, 20 May 2005 14:25:08 GMT
Server: Apache-Coyote/1.1
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://ww-
w.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:getListResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

```
xmlns:ns1="http://DefaultNamespace">
   <ns1:getListReturn href="#id0"/>
  </ns1:getListResponse>
  <multiRef id="id0" soapenc:root="0" soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/enco-
ding/" xsi:type="ns2:Document" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns2="http://xml.apache.org/xml-soap">
   <messages>
    <message>btcid1</message>
    <message>btcid2</message>
    <message>btcid3</message>
   </messages>
  </multiRef>
 </soapenv:Body>
</soapenv:Envelope>
```

4.3.2 getMessage

This is the client request to get a message:

```
POST /btp/connector/soap/service HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: 127.0.0.1
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 665

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://ww-
w.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:getXmlMessage soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://DefaultNamespace">
   <connectorLogin xsi:type="xsd:string">user1</connectorLogin>
   <connectorPassword xsi:type="xsd:string">user1</connectorPassword>
   <bizzTainerID xsi:type="xsd:string">BT...</bizzTainerID>
   <messageName xsi:type="xsd:string">ORDER</messageName>
  </ns1:getXmlMessage>
 </soapenv:Body>
</soapenv:Envelope>
```

which returns the following response

```
HTTP/1.1 200 OK
Set-Cookie: JSESSIONID=5A423337C45B737B729F63EA3B931E5D; Path=/axis
Content-Type: text/xml;charset=utf-8
Date: Fri, 20 May 2005 14:25:08 GMT
Server: Apache-Coyote/1.1
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://ww-
w.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:getXmlMessageResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://DefaultNamespace">
   <ns1:getXmlMessageReturn href="#id0"/>
  </ns1:getXmlMessageResponse>
  <multiRef id="id0" soapenc:root="0" soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/enco-
ding/" xsi:type="ns2:Document" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns2="http://xml.apache.org/xml-soap">
   <ORDER>
    ...
   </ORDER>
  </multiRef>
 </soapenv:Body>
</soapenv:Envelope>
```

4.3.3 setMessageOK

This is the client request to commit a successful download of a message:

```
POST /btp/connector/soap/service HTTP/1.0
Content-Type: text/xml; charset=utf-8
```

```
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: 127.0.0.1
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 739

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://ww-
w.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:setXmlMessageOK soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://DefaultNamespace">
   <connectorLogin xsi:type="xsd:string">user1</connectorLogin>
   <connectorPassword xsi:type="xsd:string">user1</connectorPassword>
   <bizzTainerID xsi:type="xsd:string">BT...</bizzTainerID>
   <messageName xsi:type="xsd:string">ORDER</messageName>
  </ns1:setXmlMessageOK>
 </soapenv:Body>
</soapenv:Envelope>
```

which returns the following response

```
HTTP/1.1 200 OK
Set-Cookie: JSESSIONID=F9F85BFF59EAEAB28BDE7DE34C728505; Path=/axis
Content-Type: text/xml;charset=utf-8
Date: Tue, 24 May 2005 10:26:29 GMT
Server: Apache-Coyote/1.1
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://ww-
w.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:setXmlMessageOKResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://DefaultNamespace">
   <ns1:setXmlMessageOKReturn href="#id0"/>
  </ns1:setXmlMessageOKResponse>
  <multiRef id="id0" soapenc:root="0" soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/enco-
ding/" xsi:type="ns2:Document" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns2="http://xml.apache.org/xml-soap">
   <response>
    <status-update>success</status-update>
   </response>
  </multiRef>
 </soapenv:Body>
</soapenv:Envelope>
```

4.3.4 putMessage

This is the client request to upload a message:

```
POST /btp/connector/soap/service HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: 127.0.0.1
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 6077

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://ww-
w.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:setXmlMessage soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://DefaultNamespace">
   <connectorLogin xsi:type="xsd:string">user1</connectorLogin>
   <connectorPassword xsi:type="xsd:string">user1</connectorPassword>
   <xmlMessage href="#id0"/>
  </ns1:setXmlMessage>
  <multiRef id="id0" soapenc:root="0" soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/enco-
ding/" xsi:type="ns2:Document" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns2="http://xml.apache.org/xml-soap">
   <ORDER version="1.0">
```

```
    ...
   </ORDER>
  </multiRef>
 </soapenv:Body>
</soapenv:Envelope>
```

which returns the following response

```
HTTP/1.1 200 OK
Set-Cookie: JSESSIONID=0660E3DBD970397153E2937256C484F1; Path=/axis
Content-Type: text/xml;charset=utf-8
Date: Fri, 20 May 2005 14:25:08 GMT
Server: Apache-Coyote/1.1
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://ww-
w.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:setXmlMessageResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://DefaultNamespace">
   <ns1:setXmlMessageReturn href="#id0"/>
  </ns1:setXmlMessageResponse>
  <multiRef id="id0" soapenc:root="0" soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/enco-
ding/" xsi:type="ns2:Document" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns2="http://xml.apache.org/xml-soap">
   <response>
    <upload>success</upload>
   </response>
  </multiRef>
 </soapenv:Body>
</soapenv:Envelope>
```

4.4 Java example code

The following example code makes basic use of the bizzTainer SOAP connector. The access functions are generated using Apache Axis version 1.1 from the WSDL data mentioned above.

```
package com.exxtainer.client;

import java.io.FileInputStream;
import java.io.StringWriter;
import java.net.URL;

import javax.xml.parsers.DocumentBuilderFactory;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.xml.serialize.OutputFormat;
import org.apache.xml.serialize.XMLSerializer;
import org.w3c.dom.Document;
import org.xml.sax.InputSource;

import com.exxtainer.SOAPConnector.SOAPConnector;
import com.exxtainer.SOAPConnector.SOAPConnectorService;
import com.exxtainer.SOAPConnector.SOAPConnectorServiceLocator;

/**
 * @author tre
 */
public class SimpleSOAPConnector {

  private static final Log log = LogFactory.getLog(SimpleSOAPConnector.class);

  public static void usage() {
    System.out.println("usage: SimpleSOAPConnector [keywords | options]");
    System.out.println("       interfaces basic SOAP services of the bizzTainer-hub");
    System.out.println("");
    System.out.println("       MAKE SURE THAT YOU USE AN ENCRYPTED CONNECTION SINCE YOUR PASSWORD IS
TRANSMITTED!");
    System.out.println("");
    System.out.println("            -url <url> URL of the bizzTainer SOAP services");
    System.out.println("        -login <login> connector login");
    System.out.println("   -password <password> connector password");
    System.out.println("     -message <message> message to get or set to download OK");
```

```
    System.out.println("         -btcid <btcid> bizzTainer container id");
    System.out.println("           -file <file> file with xml message to upload");
    System.out.println("");
    System.out.println("              -getList get list of messages available for download");
    System.out.println("           -getMessage get message available for download");
     System.out.println("           -setMessageOK call if message is downloaded correctly to update
server status");
    System.out.println("           -putMessage upload message");
    System.out.println("");
    System.out.println("examples:");
     System.out.println("SimpleSOAPConnector -url serviceURL -login connDd -password connPass -get-
List");
     System.out.println("SimpleSOAPConnector -url serviceURL -login connDd -password connPass -get-
Message -message ORDER -btcid BT1002..00AF.xml");
     System.out.println("SimpleSOAPConnector -url serviceURL -login connDd -password connPass -set-
MessageOK -message ORDER -btcid BT1002..00AF.xml");
     System.out.println("SimpleSOAPConnector -url serviceURL -login connDd -password connPass -put-
Message -file path/to/localfile");
    System.exit(1);
  }

  public static void main(String[] args) {

    CommandLine cl = new CommandLine(args);

    if (args.length < 4) {
      usage();
    }

    String url = cl.keyword("url");
    String login = cl.keyword("login");
    String password = cl.keyword("password");
    String message = cl.keyword("message");
    String btcid = cl.keyword("btcid");
    String file = cl.keyword("file");

    if (log.isDebugEnabled()) {
      log.debug("url=" + url);
      log.debug("login=" + login);
      log.debug("password=" + password);
      log.debug("message=" + message);
      log.debug("btcid=" + btcid);
      log.debug("file=" + file);
    }

    SOAPConnectorService service = new SOAPConnectorServiceLocator();

    try {
      /*
       * In oder to activate a SSL connection you have to do the following steps:
       *
       * 1. Open the website of your bizztainer hub
       * 2. When the web browser presents the certificate dialog to you, export the certificate to
       *    a file, e.g. 'bt.cert'.
       * 3. Create a local keystore and import the certificate you just saved:
       *    keytool -import -keystore trustedcerts.jks -alias <yourAlias>
             -storepass <yourPass> -file bt.cert
       *    When prompted for a trusted certificate answer 'yes'
       * 4. Make sure that the system property javax.net.ssl.trustStore points to the
            file "trustedcerts.jks".
       * 5. Uncomment the following three lines of code.
       *
       */
//      System.setProperty("java.protocol.handler.pkgs", "com.sun.net.ssl.internal.www.protocol");
//      java.security.Security.addProvider(new com.sun.net.ssl.internal.ssl.Provider());
//      System.setProperty("javax.net.ssl.trustStore", "trustedcerts.jks");

      if (url == null || login == null || password == null) {
        usage();
      }
      SOAPConnector client = service.getSOAPConnector(new URL(url));

      if (cl.option("getList")) {
        if (log.isDebugEnabled()) {
          log.debug("getList");
        }
        Document doc = client.getList(login, password);
```

```
        printXML(doc);
        /* check for errors in the returned document */
        /* on success a list of message/btcid is returned for single get with -getMessage */
      }
      if (cl.option("getMessage")) {
        if (log.isDebugEnabled()) {
          log.debug("getMessage");
        }
        if (message == null || btcid == null) {
          usage();
        }
        Document xmlMessage = client.getXmlMessage(login, password, btcid, message);
        printXML(xmlMessage);
        /* check for errors in the returned document */
        /* on success the message is returned */
        /* if the message is stored successfully commit with -setMessageOK */
      }
      if (cl.option("setMessageOK")) {
        if (log.isDebugEnabled()) {
          log.debug("setMessageOK");
        }
        if (message == null || btcid == null) {
          usage();
        }
        Document xmlSURes = client.setXmlMessageOK(login, password, btcid, message);
        printXML(xmlSURes);
        /* check for errors in the returned document */
        /* on success a document with 'success' is returned */
      }
      if (cl.option("putMessage")) {
        if (log.isDebugEnabled()) {
          log.debug("putMessage");
        }
        if (file == null) {
          usage();
        }
        InputSource in = new InputSource(new FileInputStream(file));
        DocumentBuilderFactory dfactory = DocumentBuilderFactory.newInstance();
        dfactory.setNamespaceAware(true);
        Document xmlMessage = dfactory.newDocumentBuilder().parse(in);

        Document xmlResponse = client.setXmlMessage(login, password, xmlMessage);
        printXML(xmlResponse);
        /* check for errors in the returned document */
        /* on success a document with 'success' is returned */
      }
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }

  }

  public static void printXML(Document document) {
    try {
      OutputFormat format = new OutputFormat(document); //Serialize DOM
      StringWriter stringOut = new StringWriter(); //Writer will be a String
      XMLSerializer serial = new XMLSerializer(stringOut, format);
      serial.asDOMSerializer(); // As a DOM Serializer
      serial.serialize(document.getDocumentElement());
      System.out.println(stringOut.toString());
    } catch (Exception e) {
      System.out.println("Service error: " + e.getMessage());
    }
  }

}
```

5 REST style interface

This section describes the REST style specific parameters of the functions described in the general section.

5.1 REST service URLs

The bizzTainer REST service URLs are listed below where <your.bizztainer.host> is the URL of the bizzTainer hub you are connected to.

http://<your.bizztainer.host>/btp/connector/rest/getList

http://<your.bizztainer.host>/btp/connector/rest/getMessage/<btcid>/<messageType>

http://<your.bizztainer.host>/btp/connector/rest/setMessageOK/<btcid>/<messageType>

http://<your.bizztainer.host>/btp/connector/rest/putMessage

<btcid> and <messageType> are the actual bizzTainer ids and message types you want to access. See the example section for this.

5.2 Access parameters

The login parameters are transferred as a HTTP header parameter according to RFC 2617 (HTTP Authentication).

The parameters *connector-login* and *connector-password* are transferred in the HTTP header as

```
Authorization: Basic connector-login:connector-password
```

where the string **`connector-login:connector-password` is base64 encoded**. Please watch that connector-login and connector-password are separated by a „:“!

A real life example would therefore look like

```
Authorization: Basic dXNlcjE6dXNlcjE=
```

Since the login parameters are transferred within the message, make sure that you use the encrypted HTTPS protocol for message exchange.

5.3 Message transfer

The following data describe the REST message transfer between the REST client and a bizzTainer hub.

5.3.1 getList

This is the client request to get the list of available messages:

```
GET /btp/connector/rest/getList HTTP/1.1
Authorization: Basic dXNlcjE6dXNlcjE=
User-Agent: Jakarta Commons-HttpClient/2.0.2
Host: 127.0.0.1
```

which returns the following response

```
HTTP/1.1 200 OK
X-Cocoon-Version: 2.1.4
Expires: Thu, 26 May 2005 11:54:29 GMT
Cache-Control: max-age=1, public
Content-Type: text/xml
Transfer-Encoding: chunked
Date: Thu, 26 May 2005 11:54:28 GMT
Server: Apache-Coyote/1.1

1c67
```

```
<?xml version="1.0" encoding="UTF-8"?>
<messages>
  <message>btcid1</message>
  <message>btcid2</message>
  <message>btcid3</message>
</messages>
```

5.3.2 getMessage

This is the client request to get a message:

```
GET /btp/connector/rest/getMessage/BT10049001-20050519-180333656-C0A800B-
C00000103F5B2029800000007.xml/ORDERRESPONSE HTTP/1.1
Authorization: Basic dXNlcjE6dXNlcjE=
User-Agent: Jakarta Commons-HttpClient/2.0.2
Host: 127.0.0.1
```

which returns the following response

```
HTTP/1.1 200 OK
X-Cocoon-Version: 2.1.4
Expires: Thu, 26 May 2005 11:54:29 GMT
Cache-Control: max-age=1, public
Content-Type: text/xml
Transfer-Encoding: chunked
Date: Thu, 26 May 2005 11:54:28 GMT
Server: Apache-Coyote/1.1

1c67
<?xml version="1.0" encoding="ISO-8859-1"?>
<ORDERRESPONSE version="1.0">
  ...
</ORDERRESPONSE>
```

5.3.3 putMessage

This is the client request to upload a message:

```
POST /btp/connector/rest/putMessage HTTP/1.1
Content-type: text/xml; charset=ISO-8859-1
Authorization: Basic dXNlcjE6dXNlcjE=
User-Agent: Jakarta Commons-HttpClient/2.0.2
Host: 127.0.0.1
Content-Length: 4948

<?xml version="1.0" encoding="UTF-8"?>
<ORDER version="1.0">
  ...
</ORDER>
```

which returns the following response

```
HTTP/1.1 200 OK
X-Cocoon-Version: 2.1.4
Expires: Thu, 26 May 2005 11:56:06 GMT
Cache-Control: max-age=1, public
Content-Type: text/xml
Transfer-Encoding: chunked
Date: Thu, 26 May 2005 11:56:11 GMT
Server: Apache-Coyote/1.1

2a1
<?xml version="1.0" encoding="ISO-8859-1"?>
<conn:connector><conn:upload-message><transform><message>ORDER</message><btid new="true">BT10049001-
20050526-135605937-C0A800BC0000010418DBF7B100000002.xml</btid><results><result step="save-uploaded-
message">success</result><result step="results">success</result><result step="map-
message">success</result><result step="validate">success</result><result name="find-partners">suc-
cess</result><result name="analyse-partners-pipeline">success</result><result step="set-status">suc-
cess</result></results></transform></conn:upload-message></conn:connector>
```

5.4 Java example code

The following example code makes basic use of the bizzTainer REST connector.

```
package com.exxtainer.client;

import java.io.File;
import java.io.FileInputStream;

import org.apache.commons.codec.binary.Base64;
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.methods.EntityEnclosingMethod;
import org.apache.commons.httpclient.methods.GetMethod;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

/**
 * @author tre
 */
public class SimpleRESTConnector {

  private static final Log log = LogFactory.getLog(SimpleRESTConnector.class);

  public static void usage() {
    System.out.println("usage: SimpleRESTConnector [keywords | options]");
    System.out.println("       interfaces basic REST services of the bizzTainer-hub");
    System.out.println("");
    System.out.println("       MAKE SURE THAT YOU USE AN ENCRYPTED CONNECTION SINCE YOUR PASSWORD IS
TRANSMITTED!");
    System.out.println("");
    System.out.println("             -url <url> URL of the bizzTainer REST services");
    System.out.println("         -login <login> connector login");
    System.out.println("   -password <password> connector password");
    System.out.println("     -message <message> message to get or set to download OK");
    System.out.println("          -btcid <btcid> bizzTainer container id");
    System.out.println("            -file <file> file with xml message to upload");
    System.out.println("");
    System.out.println("               -getList get list of messages available for download");
    System.out.println("           -getMessage get message available for download");
    System.out.println("         -setMessageOK call if message is downloaded correctly to update
server status");
    System.out.println("           -putMessage upload message");
    System.out.println("");
    System.out.println("examples:");
    System.out.println("SimpleRESTConnector -url serviceURL -login connDd -password connPass -get-
List");
    System.out.println("SimpleRESTConnector -url serviceURL -login connDd -password connPass -get-
Message -message ORDER -btcid BT1002..00AF.xml");
    System.out.println("SimpleRESTConnector -url serviceURL -login connDd -password connPass -set-
MessageOK -message ORDER -btcid BT1002..00AF.xml");
    System.out.println("SimpleRESTConnector -url serviceURL -login connDd -password connPass -put-
Message -file path/to/localfile");
    System.exit(1);
  }

  public static void main(String[] args) {

    CommandLine cl = new CommandLine(args);

    if (args.length < 4) {
      usage();
    }

    String url = cl.keyword("url");
    String login = cl.keyword("login");
    String password = cl.keyword("password");
    String message = cl.keyword("message");
    String btcid = cl.keyword("btcid");
    String file = cl.keyword("file");

    if (log.isDebugEnabled()) {
      log.debug("url=" + url);
      log.debug("login=" + login);
      log.debug("password=" + password);
      log.debug("message=" + message);
      log.debug("btcid=" + btcid);
      log.debug("file=" + file);
    }
```

```
    try {
      /*
       * In oder to activate a SSL connection you have to do the following steps:
       *
       * 1. Open the website of your bizztainer hub
       * 2. When the web browser presents the certificate dialog to you, export the certificate to
       *    a file, e.g. 'bt.cert'.
       * 3. Create a local keystore and import the certificate you just saved:
       *    keytool -import -keystore trustedcerts.jks -alias <yourAlias>
       *       -storepass <yourPass> -file bt.cert
       *    When prompted for a trusted certificate answer 'yes'
       * 4. Make sure that the system property javax.net.ssl.trustStore points to the
       *    file "trustedcerts.jks".
       * 5. Uncomment the following three lines of code.
       *
       */
//      System.setProperty("java.protocol.handler.pkgs", "com.sun.net.ssl.internal.www.protocol");
//      java.security.Security.addProvider(new com.sun.net.ssl.internal.ssl.Provider());
//      System.setProperty("javax.net.ssl.trustStore", "trustedcerts.jks");

      if (url == null || login == null || password == null) {
        usage();
      }
      String toEncode = login + ":" + password;
      String encodedContent = new String(Base64.encodeBase64(toEncode.getBytes()));
      String authStr = "Basic " + encodedContent;

      if (cl.option("getList")) {
        GetMethod get = new GetMethod(url);
        get.setRequestHeader("Authorization", authStr);
        HttpClient httpclient = new HttpClient();
        int result = httpclient.executeMethod(get);
        System.out.println("Response status code: " + result);
        System.out.println("Response body: ");
        System.out.println(get.getResponseBodyAsString());
        get.releaseConnection();
        /* check for errors in the returned document */
        /* on success a list of message/btcid is returned for single get with -getMessage */
      }
      if (cl.option("getMessage")) {
        if (message == null || btcid == null) {
          usage();
        }
        GetMethod get = new GetMethod(url + "/" + btcid + "/" + message);
        get.setRequestHeader("Authorization", authStr);
        HttpClient httpclient = new HttpClient();
        int result = httpclient.executeMethod(get);
        System.out.println("Response status code: " + result);
        System.out.println("Response body: ");
        System.out.println(get.getResponseBodyAsString());
        get.releaseConnection();
        /* check for errors in the returned document */
        /* on success the message is returned */
        /* if the message is stored successfully commit with -setMessageOK */
      }
      if (cl.option("setMessageOK")) {
        if (message == null || btcid == null) {
          usage();
        }
        GetMethod get = new GetMethod(url + "/" + btcid + "/" + message);
        get.setRequestHeader("Authorization", authStr);
        HttpClient httpclient = new HttpClient();
        int result = httpclient.executeMethod(get);
        System.out.println("Response status code: " + result);
        System.out.println("Response body: ");
        System.out.println(get.getResponseBodyAsString());
        get.releaseConnection();
        /* check for errors in the returned document */
        /* on success a document with 'success' is returned */
      }
      if (cl.option("putMessage")) {
        if (file == null) {
          usage();
        }
        PostMethod post = new PostMethod(url);
        File input = new File(file);
        post.setRequestBody(new FileInputStream(file));
```

```
        if (input.length() < Integer.MAX_VALUE) {
          post.setRequestContentLength((int) input.length());
        } else {
          post.setRequestContentLength(EntityEnclosingMethod.CONTENT_LENGTH_CHUNKED);
        }
        post.setRequestHeader("Content-type", "text/xml; charset=ISO-8859-1");
        post.setRequestHeader("Authorization", authStr);
        HttpClient httpclient = new HttpClient();
        int result = httpclient.executeMethod(post);
        System.out.println("Response status code: " + result);
        System.out.println("Response body: ");
        System.out.println(post.getResponseBodyAsString());
        post.releaseConnection();
        /* check for errors in the returned document */
        /* on success a document with 'success' is returned */
      }
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
  }
}
```